Adaptive Sampling for Low Latency Vision Processing

David Gibson[1], Henk Muller[2], Neill Campbell[1] and David Bull[1]

[1]Computer Science, University of Bristol, UK
[2]XMOS Ltd, Bristol, UK

Abstract. In this paper we describe a close-to-sensor low latency visual processing system. We show that by adaptively sampling visual information, low level tracking can be achieved at high temporal frequencies with no increase in bandwidth and using very little memory. By having close-to-sensor processing, image regions can be captured and processed at millisecond sub-frame rates. If spatiotemporal regions have little useful information in them they can be discarded without further processing. Spatiotemporal regions that contain 'interesting' changes are further processed to determine what the interesting changes are. Close-to-sensor processing enables low latency programming of the image sensor such that interesting parts of a scene are sampled more often than less interesting parts. Using a small set of low level rules to define what is interesting, early visual processing proceeds autonomously. We demonstrate system performance with two applications. Firstly, to test the absolute performance of the system, we show low level visual tracking at millisecond rates and secondly a more general recursive Baysian tracker.

1 Introduction

There is increasing interest in low cost computer vision systems with a wide range of applications including gesture based user interfaces, surveillance, automotive systems and robotics. As the complexity of consumer, sensing and military systems increase the demands on energy resources becomes critical for high-level computing performance. Vision systems are proving to be extremely valuable across a range of applications and to be able to efficiently process visual information offers a huge advantage in the functionality of such systems.

Traditional computer vision systems typically consist of a camera continually capturing and transmitting images at a fixed frame rate and resolution with a host computer sequentially processing them to obtain a result such as the trajectory of a moving object. A major drawback of this pipeline is that large amounts of memory are required to store the image data before it is processed, especially as frame rate and image resolution increase. Additionally large amounts of the image data is transmitted to the host for processing regardless of the amount of information contained in this data. In the case of object tracking, computer vision algorithms work towards creating a concise description such as, 'a group of pixels at a certain location is moving in a particular way'. Often the object is

relatively small compared to the whole image and the background maybe static. In cases like this, the traditional computer vision processing pipeline could be considered as being highly inefficient as large amounts of image data are being captured, transmitted to the host, stored in memory and being processed on a per-pixel basis while most of the visual information comes from a small number of changing pixels. In such cases most of the image data is discarded as it contains no useful information.

In the case of a scene with an object moving across a static background most of the image data changes very little while some pixel areas might change rapidly or move a different speeds. The fixed temporal sampling rate of standard camera systems cannot take this into account and artifacts such as motion blur and temporal incoherence are introduced. These artifacts consequently confound down stream processing necessitating ever more complex computer vision algorithms to overcome these imaging effects.

A significant problem with digital video capture is that of temporal quantization. Given a finite set of resources, digital video capture proceeds by sequentially sampling frames at *fixed* temporal rates and spatial resolution within a range of luminance in a non-interactive passive manner. Biological systems proceed very differently; unable to sequentially process entire views, selective scene sampling is performed using a combination of eye movements. In Rucci et.al. [1–5] it is shown that a number of strategies exist for visual sampling in human vision depending on the task being carried out. A human eye is constantly moving in order to avoid fading, the loss of sight due to a lack of change on the retina. As well as head movement, eye movements include saccades, micro-saccades and drift. These movements enhance and stabilize the binocular view allowing the process of foveation to create a highly detailed perception capable of difficult tasks such as threading a needle.

Modern high speed cameras are capable of capturing images at thousands of frames a second and can have dedicated processing modules close to the sensor. In [6] the wing of a fly was tracked using regions of interest (ROI) at 6000Hz. The system used edge detection to analyse the shape and motion of the fly's wing via feedback from a tracker which predicted the next ROI. One problem with such systems is they are rigid in their FPGA based design, are task specific and highly implementation and environment dependent.

In order to increase the flexibility and efficiency of high-level downstream processing, image sensor design companies are developing devices that can compute interest points and local descriptors in silicon [7]. Other silicon devices include the artificial retina [8]. The artificial retina is being investigated in a number of contexts, one of which is 'event-based stereo vision in real-time with a 3D representation of moving objects' [9]. The low latency of this device is of particular interest. However, there is no illumination detail provided. This is overcome in a hybrid system that includes a traditional digital camera system to investigate selective attention or saliency for real-time active vision [10].

In a keynote speech Ed Dowski [11], lead for new technologies at OmniVision CDM Optics, Inc. suggested that:"An important class of future imaging systems,

in our view, will be Application-Specific Imaging Systems (ASIS). These imaging systems will not be general purpose in that they are meant primarily for human viewing, but will be specialized systems that capture and communicate specific types of information depending on the particular application."

The central hypothesis of this paper is that low latency visual sampling can provide a framework for solving many challenging real world vision problems. The proposed system has characteristics similar to those of several compressive sensing methods [12]. Non-linear visual sampling in the spatial and temporal domains followed by image reconstruction of whole image sequences is a popular research avenue. However, current compressive sensing techniques generally involve highly specialised and expensive components with the results being reconstituted using time consuming and computationally demanding algorithms rendering them difficult to apply to practical real-world problems.

In this paper we are particularly interested in exploiting spatiotemporal redundancy **and** the low latency control offered by close-to-sensor processing through the use of non-linear visual sampling and **piecewise visual processing**. By exploiting spatiotemporal redundancy, high speed imaging can be accomplished without increasing bandwidth while reducing errors introduced by temporal quantization. Low latency enables software pipelines that can be switched so as to adapt to changes in visual input and be posed as a **functional visual sampling** problem. The proposed system and associated algorithms are strictly real-time in the sense that the capture and image processing relationship is directly linked and interdependent. An advantage of the proposed systems is that a broad range of traditionally hard or impossible vision based processing tasks can be addressed within a novel, cost and energy efficient framework. We propose a highly programmable visual sampling approach for application specific imaging pipelines (ASIP), that can provide output for machine vision systems as well as human observers.

1.1 Hardware system

Central to the system design is the XMOS microprocessor and the ability to reprogram image sensor parameters with very low latency. The processor allows a direct connection to an image sensor, has no operating system, does not use interrupts and supports concurrent threads within a parallel architecture. The XMOS[1] XCore is a multi-threaded processing component with instruction set support for communication, I/O and timing. Thread execution is deterministic and the time taken to execute a sequence of instructions can be accurately predicted. This makes it possible for software executing on an XCore to perform many functions normally performed by hardware, especially DSP and I/O.

To investigate the exploitation of spatiotemporal redundancy via piecewise visual processing a development board has been designed and built. Figure 1 shows the layout of the latest visual processing system. The design is such that pixel information is read in from an image sensor a one end of the pipeline and

[1] www.xmos.com

then processed into higher and higher representations as data passes through the system. Communications via ethernet and RAM is available at the far end of the pipeline. The pipeline can be configured in software and feedback to the image sensor control thread can be provided at any stage of processing. It should be noted that the original sampled pixel information need never be lost and in the simplest configuration the system behaves as a standard camera. With nine xcores in total 4500MIPS of processing across 36 concurrent threads is available for processing. XLinks provide fast inter core bi-directional communications and extensive GPIO is available.

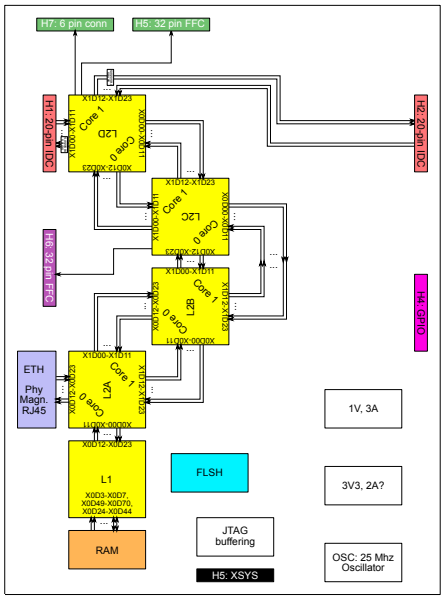

Fig. 1. Block diagram for the latest hardware design. It consists of four dual core and a single core processors, 128MB SDRAM, ethernet phy, approximately ten smaller chips (flash, buffers, oscillator, reset supervisors, etc), one FFC connector for the image sensor (H5 and H7), one FPC connector for the LCD (H6), male and female 16 pin IDC headers for XLinks (H1 and H2). XLinks enable multiple boards to be connected with each other and have a bandwidth of 320Mb/s. XLinks are also what form the backbone of the pipeline connecting the processor in series from sensor input at L2D to ethernet and RAM on L2A and L1 respectfully. Extensive GPIO is available on header H4.

Figure 2 shows an advanced system layout that could be implemented by the design in figure 1. For the work described in this paper less elaborate subsystem designs have been used. The minimal configuration consists of an image sensor connected to an XMOS processor and software running on a single xcore; one thread being used to read in pixels and control the sensor and a further

two threads to run a UDP ethernet transmitter. The architecture is highly programmable; if several xcores are linked a wide range of processing and feedback designs can be implemented.

The underlying imaging used in this paper is, rather than capture an entire image and then process the pixels at a standard frame rate, regions of interest (ROI) are captured and processed at high frame rates. An advantage gained using this approach is that an ROI can be processed in the time it takes to expose the next ROI. The key advantage of this approach is that areas of an image that have interesting changes occurring in then can be sampled more often than in image areas where no changes are occurring. Throughout this paper the pixel depth was set to 8 bits and ROI were set to 64 by 40 pixels, the sensor resolution was set to 640 by 480 pixels and 2 by 2 pixel binning was used to give an effective image resolution of 320 by 240 pixels. Each xcore has 64KB of on chip memory, all of the follow experiments use only this memory and were carried out using two quad-core XMOS processors.

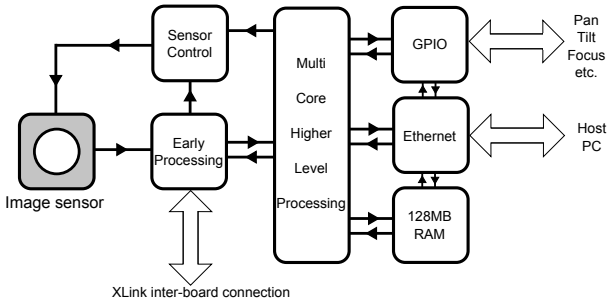

Fig. 2. A system diagram for a fully functioning processing pipeline. Early processing and sensor control execute in separate dedicated threads close to the sensor. Higher level processing and communications occurs in many threads over several cores and processors.

In the following section two applications for low level visual tracking are described. They are based on the system layout of figure 2. As such the two trackers can be seen as running in parallel with each other. If there are no interesting changes with respect to the millisecond tracker pixel data is passed on to the higher level Baysian tracker. The Baysian tracker then directs the re-programming of the image sensor according to what it determines as the next interesting ROI to sample. However, if the millisecond tracker does detect pixels that are interesting to it, it now overrides higher level processing and proceeds of its own accord. When the millisecond tracker no longer detects interesting changes control of the sensor re-programming is handed back to the higher level processing system. These two application are now described in more detail.

1.2 Experiments and Results

After initialization the system proceeds by repeating the following; program the ROI position on the sensor, exposing the ROI, read in ROI pixels from sensor, process ROI pixels. After a short period of time temporal differences can be computed, i.e. the difference between two spatially coherent but temporally offset ROI. This temporal differencing provides the basis for low level event detection and tracking. The sensor can be run in two modes; free running or triggered. The design in figure 1 allows for both. However, the hardware used in this paper could only function in free running mode. To understand the performance and limitation of the system the sensor was run at full speed, table 1 describes the low level timings and identifies when and how processor cycles are being used.

Frame Grab	Data Send	Frame Update	Frame Delay	FPS
42384	2042	46480	61296	964
42384 [a]	2042	32544	61296	964
42384 [b]	32	32538	61296	964
42384 [c]	32	4682	9455	1928
42384 [d]	32	34	9455	1928

Table 1. Timing information in CPU cycles at 100MHz with frame and row delay at zero and sensor clock at 25MHz. One frame is dropped in the first three rows with overall frame time being $\sim$ 103000 cycles (frame grab plus frame delay, 964fps). The time between row read-ins is 816 cycles.

[a] No display window update. OK if display columns remain constant.
[b] No data send.
[c] No window update, with display window update.
[d] No update at all.

In table 1 timing information in CPU cycles at 100MHz with frame and row delay at zero and sensor clock at 25MHz is given. The order of processing is as follows; the frame ready pin is pulled high (this is when timing starts), pixels are read in on a line by line basis, the pixel data is transmitted to the ethernet transmitter thread, the ROI position is updated via I2C and the next exposure begins. This cycle is repeated over the whole sensor surface, returning back to $(0, 0)$ after each 320 by 240 composite image is read if the sensor windows are updated. From table 1 it is clear that transmitting the pixel data on to the next thread takes roughly two thousand clock cycles. The display window is the sensor width by ROI height region that the sensor exposes and the window update is the region of pixels that the sensor reads out. The time it takes to update the display window does not effect the FPS value, however updating the read out

window involves programming over twice as many registers. Updating the read out window with or without updating the display window takes longer than the exposure time, consequently a whole frame is dropped before the frame ready pin is pulled high. With no window updates and no data send the absolute performance is demonstrated; it take 42384 cycles to read in the pixel data and 9455 cycles to expose the sensor giving an absolute frame rate of 1928FPS ($100000000/(42384+9455)$). If the sensor windowing updates take less than the exposure time then a frame is not dropped.

Millisecond tracker The above timings are with respect to a single thread running at 100MHz and with the delay between rows read-ins (816 cycles) there remain roughly 50000 cycles for processing the 64 by 40 ROI. To perform single ROI low level tracking a background model is built for each incoming ROI. This consists of two histograms, one for the maximum values of each row and one for the maximum of each column. If a single peak over a certain threshold exists a point of interested is considered as detected. If in the next ROI a similar point exists tracking begins and the x and y offsets between the two peaks are used to initialize a predictive tracker. The mean value of the previous and current prediction is used to estimate where the point in the next ROI will be. The position of the ROI on the sensor is updated and the process is iterated until the interest point is lost. Figure 3 show the laser point stimulus. The motion is so fast that at 30fps the light is smeared across the exposures. In figure 4 some example frames from the tracking result are shown, behind the text there is a bright point light. It should be noted that there is not a direct one to one match of fields of view or temporal synchronization between the images shown in Figures 3 and 4 as the different sensors are in different positions and are not fully aligned.

Fig. 3. The the laser point stimulus. The motion is so fast that at 30fps the light is smeared across the exposures. The point light is moved using a servo that has a maximum rotation speed of 0.16s for 60 degrees.

Baysian tracker If the above criteria for a point of interest is not met, ROI pixel data is transmitted on to a more complex tracking system. This is now described in detail. Given an image sensor surface, $\mathbb{S}$, with resolution $[X, Y]$ a spatiotemporal volume is described as $v(x, y, t)$ where $x \in [1, X]$ and $y \in [1, Y]$ are the row and column coordinates respectively and t is the temporal coordinate. A non-overlapping rectangular grid is defined as $g_o \in G \equiv \mathbb{S}$ and g_s is the set

Fig. 4. Fast point tracker. Behind the text there is a point light. The top row of numbers shows the x and y positions within the ROI of the max pixel value. The second row shows the predeicted position of the point ligth according to the first order tracker. The third row shows the x value of the predictive tracker.

of all possible rectangles that belong to G. The image sensor is programmed by registers that can be set, on a sub-image by sub-image basis, enabling selective sampling of G with variable intervals of t. Initially, over all g_o we sample sub-images, $s(x_s, y_s)$, where x_s and y_s are the rows and columns of each sub-image, generally $x_s < (X/4)$ and $y_s < (Y/4)$. There are N_o non-overlapping sub-images in g_o that cover $\mathbb{S}$ and as each sub-image is captured some sparse feature vector representation, $\boldsymbol{f}$, of each sub-image is computed. After $t(N_o + 1)$ samples, sub-images s_1 and s_{N_o+1} are compared to determine if any changes have occurred. Any metric can be used to determine if and how the samples might have changed, the simplest is a difference, $d_{g_o(1)} = (\boldsymbol{f}_{N_o+1} - \boldsymbol{f}_1)$. When $t(2 \times N_o)$ samples have been captured a multi-modal distribution of differences across the extent of G is computed as:

$$p(d)_{t=0} = pdf(d) = \int_{g_o} \frac{1}{\sqrt{2\pi}} e^{-d^2/2} \tag{1}$$

Equation 1 initializes the system; if $p(d) = 0$, no changes in the pairs of sub-images have occurred otherwise $p(d)$ is proportional to the magnitude of change according to the feature description and metric used. $p(d)$ is updated with each new differential observation, d, in a manner similar to a large class of algorithms that include sequential Baysian filtering, Kalman filters, particle filters, HMMs, etc.

$$p(d)_t = \int_{g_o} \frac{1}{\sqrt{2\pi}} e^{-(p(d)_{t-1} + d_t)^2/2} \tag{2}$$

So far, x_s and y_s belong to g_o and δt is constant. The proposed algorithm now proceeds by re-sampling $p(d)$ such that a new set of sub-images, s_{g_p}, where $g_p \in g_s$, predict likely visual changes at some time in the future:

$$(g_p, t_{g_p}) \leftarrow p(d) \tag{3}$$

The algorithm proceeds to iterate over the Equation 2 and 3 effectively tracking visual changes that are 'interesting' according the feature set description and difference metric. The above description represents the simplest formulation of the proposed system, more complex formulations easily fit within the

same framework. Equation 3 provides the basis for the hypothesis of this paper; δt_{g_p} and the number of sub-images, s_{g_p} are not fixed. There are several interesting consequences of this; firstly no whole images exist in the traditional sense, secondly there is no fixed frame rate, sub-images are captured at different spatial location and at different temporal rates depending on changes in the visual scene. An area of a scene where little or no change occurs gets sampled infrequently and the δt between corresponding temporal sub-images will be relatively large. An area that changes a lot and rapidly will be sample frequently and δt will tend towards its minimum. In the current and proposed hardware design $\min(\delta t) \leq 0.5ms$. As fewer sub-images are sampled more frequently there is no significant difference in bandwidth compared to the bandwidth of standard frames rates and resolutions. It should be noted that the original sampled pixel information need never be lost and offers the potential for compressive sensing or other more standard techniques to be implemented further down the pipeline. Figure 5 shows the overall composition of the higher level tracking system. Figure 6 shows the individual components of the image processing pipeline and figure 7 shows selected frames of an object being tracked.

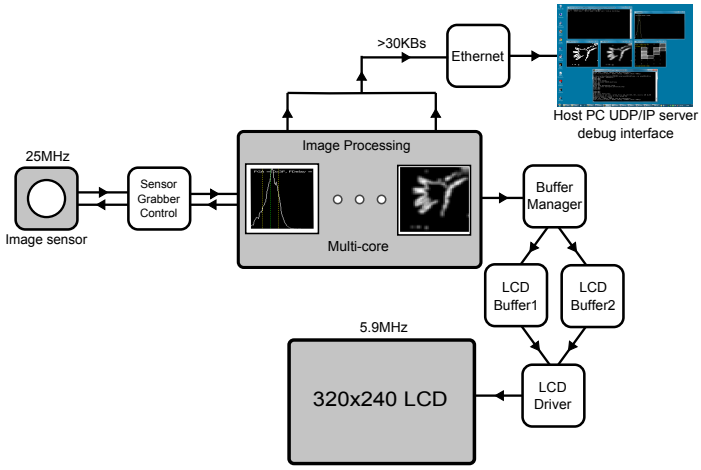

Fig. 5. The composition of the recursive Baysian tracker system. The full system is shown including a host PC that enables the internal states of the pipeline to be visualised.

1.3 Discussion

The work presented in this paper is in its infancy and the authors expect to be able to create much more advanced systems as dedicated hardware and higher quality image sensors become available. We will research and develop multiple pixel processing pipelines that implement low latency detection and tracking,

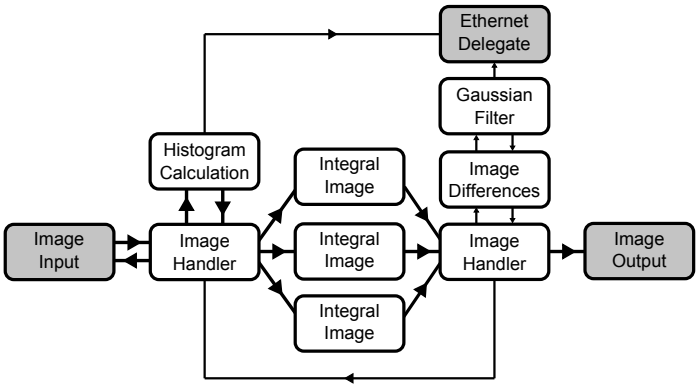

Fig. 6. The the individual components of the image processing pipeline. Each units represents a single concurrent thread running on a quadcore XMOS processor.

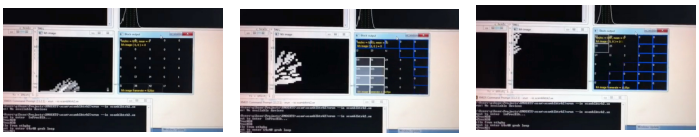

Fig. 7. Adaptive camera tracker, selected frames of an object being tracked. Blue rectangles are sensor surface regions that are momentarily being less frequently sampled.

autonomous stereo alignment and higher level vision processing. Multiple spatiotemporal resolutions will be used to direct focus of attention and stabilize vision algorithms. A major potential of the work is to investigate image sampling strategies given a particular stimulus and/or task. We will learn the underlying rules that enable the system to change its mode of operation. We will investigate autonomous pan, tilt and focus so as to provide continuously changing perspectives of any given visual stimulus. Sensor-processors will be able to change their line of sight automatically based on low level rules for tasks such as; follow and focus on the largest moving object in a scene. To understand how a stereo pair of sensors might automatically configure themselves is particularly interesting, being able to move and focus independently allows a sensor pair to optically search over the depths of multiple objects within a scene.

Initial work on an adaptive processing pipeline for low level visual tracking has been presented, more advanced tasks could include; 'track a single object at a frame rate that minimizes motion blur', 'track the depth of the most interesting moving object in the scene', 'generate a super resolution snap shot of the most interesting object within a scene' or 'compute optical flow if the whole scene changes rapidly', etc. These tasks can be combined such that as a scene changes the most appropriate mode of pipeline operation is selected.

1.4 Conclusions

Traditional digital imaging is generally a passive process whereby images of fixed size and frame rates are captured regardless of what is occurring in the scene. Understanding motion cues is often made easier by increasing frame rate. However, this greatly increases the amount of data that needs to be transmitted and processed. Additionally, reprogramming a cameras image sensor often takes a number of frames leading to a latency between what an artificial system has processed and what the next image content might be. We have shown that by adaptively sampling visual information with respect to what is occurring in a scene, the performance of low level vision systems can be improved without increasing bandwidth. By having close-to-sensor processing, image regions can be captured and processed very rapidly. If spatiotemporal regions have little useful information in them they can be discarded without further processing. Spatiotemporal regions that contain 'interesting' changes are further processed to determine what the interesting changes are. Close-to-sensor processing enables low latency programming of the image sensor such that interesting parts of a scene are sampled more often than less interesting parts. Using a small set of low level rules to define what is interesting, early processing proceeds autonomously with very low latency.

The presented hardware design offers a cost effect high frame rate computational camera. As image processing is carried out in a piecewise manner a traditional frame store is not required. This in turn reduces the complexity of the system. The deterministic and parallel nature of the XMOS architecture allows for efficient and flexible visual processing pipeline designs.

1.5 Future work

It should be noted that the default behaviour of the proposed system is that of a standard camera and original pixel information need never be lost. The proposed system is a computation camera capable a wide range of functionality. Multiple systems can be connected within the plug and play design to create multi-sensor systems. A four part system could consist of a monochrome stereo pair, a colour sensor and an IR sensor all with fast interconnections and 18000MIPS of parallel computing resources. With the existing design this would cost less than 1000USD, be the size of a small laptop and interface via a standard ethernet connection. In future work we plan to build a much smaller and more powerful design. One motivation for this is to be able to make the system more widely accessible to the vision and robotics community. Figure 8 shows the latest hardware design and roughly mirrors the layout described in figure 1.

1.6 Acknowledgements

This work was supported by the CDE Futures and Innovation Domain, DSTL CDE24704.

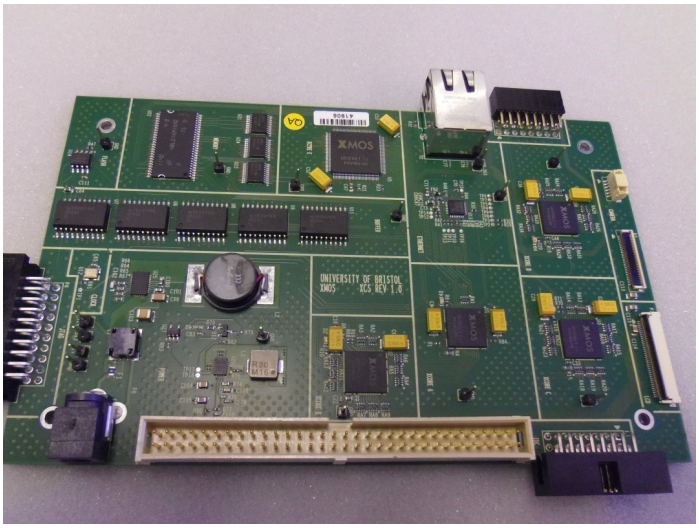

Fig. 8. The latest hardware design which is based on the block diagram layout described in figure 1.

References

1. M. Rucci, R. Iovin, M. Poletti, and F. Santini: Miniature eye movements enhance fine spatial detail. Nature (2007)
2. F. Santini, R. Nambisan and M. Rucci: Active 3d vision through gaze relocation in a humanoid robot. International Journal of Humanoid Robotics (2009)
3. H. Ko, M. Poletti and M. Rucci: Microsaccades precisely relocate gaze in a high visual acuity task. Nature Neuroscience (2010)
4. M. Poletti, C. Listorti and M. Rucci: Stability of the visual world during eye drift. Journal of Neuroscience (2010)
5. M. Poletti and M. Rucci: Eye movements under various conditions of image fading. Journal of Vision (2010)
6. C. F. Graetzel, S. N. Fry and B. J. Nelson: A 6000 hz computer vision system for real-time wing beat analysis of drosophila. International Conference on Biomedical Robotics and Biomechatronics **1** (2006)
7. G. Kirsch: Interest point and local descriptor generation in silicon. International Solid-State Circuits Conference (2012)
8. P. Lichtsteiner, C. Posch and T. Delbruck: An 128x128 120db 15us-latency temporal contrast vision sensor. IEEE Journal Solid State Circuits (2007)
9. S. Shraml and A. N. Belbachir: A spatio-temporal clustering method using real-time motion analysis on event-based 3d vision. International Conference on Computer Vision and Pattern Recognition (2010)
10. D. Sonnleithner and G. Indiveri: A neuromorphic saliency-map based active vision system. International Conference on Information Sciences and Systems (2011)
11. E. Dowski: A new paradigm for future application-specific imaging systems. International Conference on Computational Photography (2011)

12. D. Reddy, A. Veeraraghavan and R. Chellapa: P2c2: Programmable pixel compressive camera for high speed imaging. International Conference on Computer Vision and Pattern Recognition (2011)